2018

An Algorithmic Approach to Creating Effective Study Groups Using a Smart Phone App

Kelvin J. Rosado-Ayala
Georgia Southern University

An Algorithmic Approach to Creating Effective Study Groups Using a Smart Phone App

An Honors Thesis submitted in partial fulfillment of the requirements for Honors in Computer Science.

By
Kelvin Rosado-Ayala

Under the mentorship of Dr. Andrew Allen

ABSTRACT

For many students entering college, meeting new people and studying are a common struggle. Study groups are generally recommended, especially if the groups are comprised of members with complementary personality traits. But the challenge still remains, how do freshmen or transfer students find and form these heterogeneous study groups. In order to help alleviate this issue, an Android application was developed to automatically create study groups for students. Using basic information provided by students upon registration, the algorithm is able to automatically find matching group members. The application was designed using an agile life cycle model over the course of a year. Once development was completed, a study was conducted using the application which analyzed the resulting data and found it to be effective.

Thesis Mentor: _____

Dr. Andrew Allen

Honors Director: _____

Dr. Steven Engel

April 2018
Department of Computer Sciences
University Honors Program
Georgia Southern University

Table of Contents

Acknowledgements

Firstly, I would like to thank my advisor Dr. Andrew Allen for agreeing to guide me through the process of completing my thesis. His advice was invaluable over the course of the project.

I would also like to thank Georgia Southern University and the University Honors Program for the opportunities that I have been given over the past four years. I am so grateful for the many doors that have been opened for my career.

In addition, I would like to thank my parents for everything they have provided me and for all the help they have given.

Additionally, I would like to thank my friend Morgan Gallahue for helping me create and distribute an interest survey for my app.

Lastly, I would like to thank my friends and family for all the support, laughs, and guidance that they have provided.

Introduction

Studying is one of the unavoidable activities in a student's life, especially for those enrolled in a college or university. While some students prefer to study alone, many prefer to study in groups, which research has shown allows students to "develop critical skills and construct common knowledge" [1]. Additionally, a study on the impact of study groups on academic performance found that "core term grades are driven by heterogeneity in group ability" [3], that is they found that membership should consist of people with differing abilities. Before these benefits can be obtained, the study group must first be formed. One common method for this is talking to classmates or roommates. For freshman and transfer students, it may take some time to get familiar with classmates. For those who are introverted, it may be impossible to initiate a study group. However, in this modern era of using smartphones for everything from ordering food to dating, there should be an app for creating study groups as well.

Although there are various apps on the market branded as being study group apps, namely BuddyUp and StudyRoom, they all have the same issue; They only provide an interface for communication and expect the users to have already formed groups. To my knowledge, no currently existing app, automates the process of creating study groups for its users. This lack of automation served as the inspiration for this project.

The main objective when designing the app was ensuring that it would be able simplify the process of creating study groups. Freshmen students were the target demographic as they are less likely to know their classmates and thus would have a more difficult time finding suitable members for a study group.

Algorithm

In order to devise an algorithm that would create effective study groups, multiple sources were checked to determine which characteristics are present in effective groups. One highly referenced source, which conducted a study on the impact of study groups and roommates on academic performance, found that for study groups "core term grades are driven by heterogeneity in group ability" [3]. Due to this finding, emphasis was placed on creating study groups with people of differing abilities in order to make the groups heterogeneous. It was also determined that heterogeneity would be determined based on four characteristics; motivation, contribution, self-direction, and preparation.

Motivation is a measure of how much the student wants to learn and retain the material in the course. Contribution is a measure of how often the student participates in productive conversation when meeting with a study group. Self-direction is a measure of how much effort the student puts into learning course material when not in a classroom setting. Preparation is a measure of how much effort the student puts in before a study group meeting to make sure that the meeting is productive.

By focusing on these personality characteristics, as opposed to specific academic subject knowledge, the same criteria can be used to create study groups for all courses thus making the algorithm general purpose. Additionally, it was determined that a study group with a size of four students is effective as it is large enough to have a wide breadth of knowledge amongst its members while still being small enough that each member's contribution is significant. With these criteria determined, development of the algorithm was ready to begin.

A python script was written in order to implement the algorithm. To create study groups for a certain course, the script first imports the list of students taking the specified course and stores that information as a dictionary to allow for efficient retrieval. In order to use this data for the algorithm, each student is represented as a point in four-dimensional space with each axis representing a specific characteristic. A three-dimensional simplified representation of sample data is shown below in figure 1.

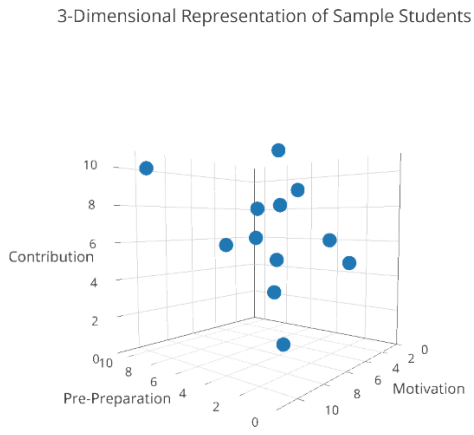


Figure 1: Three-Dimensional Representation of Sample Students

With the user population reduced to a set of four-dimensional points, the problem of creating heterogeneous is reduced to a problem of finding the farthest points. To accomplish this, the distance formula is used to calculate a list of distances between points, which is then stored for future use. With the list of distances calculated, the pair of points with the farthest distance is grouped together as the root of a study group. These points are then removed from the set and the next farthest points are grouped together to form another root. This process is repeated until the set of n points is reduced to a set of $\frac{n}{2}$ study group

roots. Following this, the root comprised of the two farthest points and the root comprised of the two closest points are combined to create a study group of four members. This process is repeated for the other roots until all students are placed into a study group. By proceeding in this order, groups are created with students covering each other's weaknesses. The group creation algorithm is illustrated in figure 2. The resulting study groups are then sent to the server to be stored.

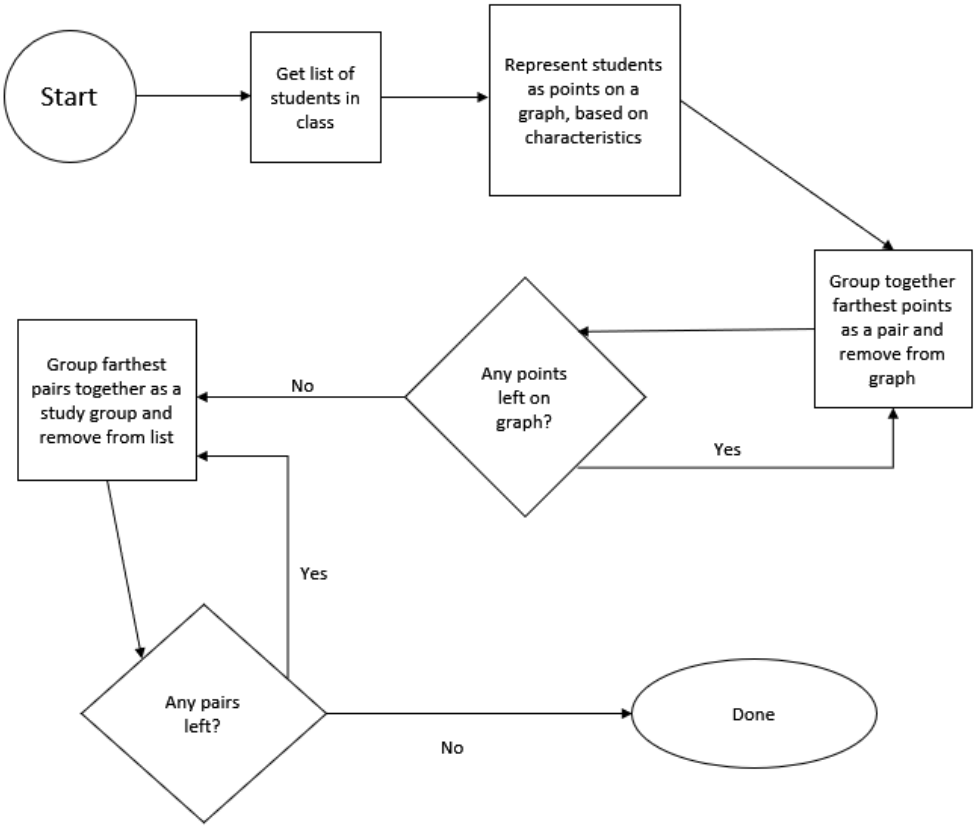


Figure 2: Group Creation Flowchart

Methods

The first step before beginning development of the app was gauging interest in the idea. To accomplish this a survey was created on Google Forms and distributed to students living on campus. The survey contained four questions; year in college (freshman,

sophomore, junior, or senior), honors status (in honor's program or no), interest level in an app that automatically creates study groups, and additional comments. The question asking about interest level in the idea asked participants to give a numerical ranking of their interest "in an app that automatically creates study groups" from one to ten, with one being no interest and ten being extreme interest. The results of the survey are summarized below in figures 3 and 4. The full results are available in Appendix A.

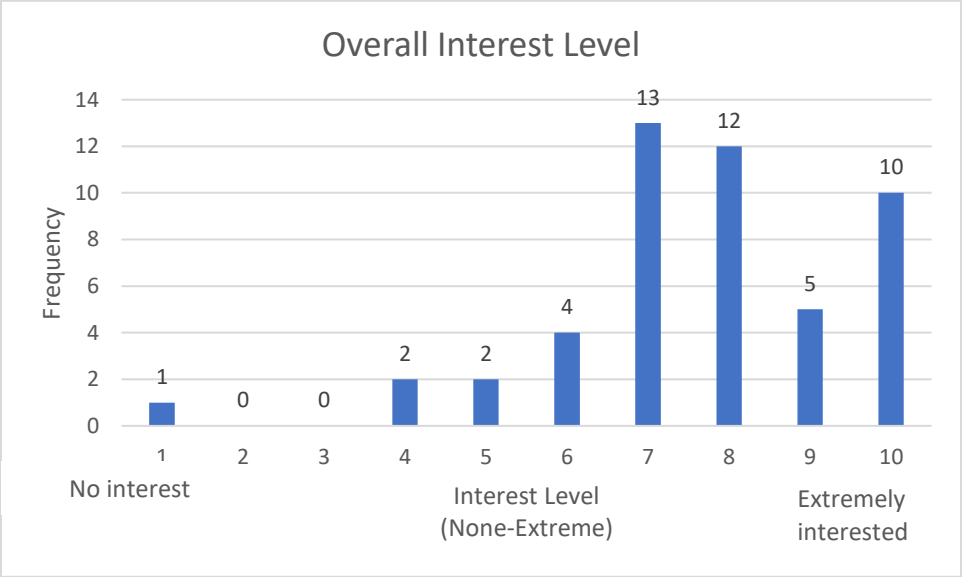


Figure 3: Overall Interest Level

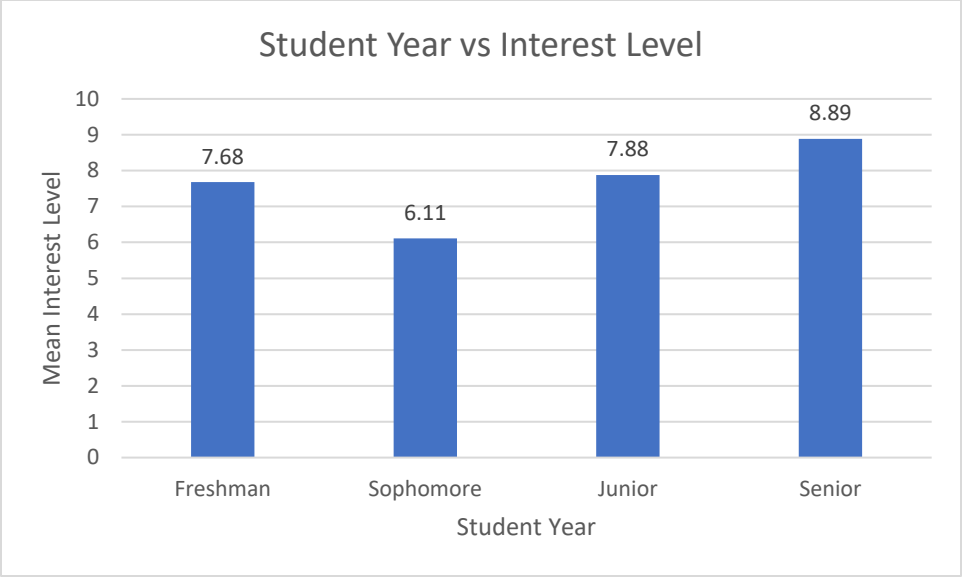


Figure 4: Year vs Interest Level

As shown in Figure 1, the responses were mostly positive with the mean interest level being 7.65 across all responses. An unexpected finding in the results was that sophomores had the overall lowest interest while seniors had the highest interest. This contradicted the hypothesis that freshman would have the highest interest as they would have the least amount of connections to create study groups on their own. With results being mostly positive, the next step was to determine the list of features to be included I the app.

A list of desired features was obtained through a combination of comments left on the survey and taking to students around the university. Among these requirements were a course search and messaging to group members. With this list of features, and an idea for the algorithm, the next step was to obtain a server to host the database as well as run the algorithm.

Once an AWS EC2 instance was obtained and set up, the necessary software was installed. The biggest of these were Python 3 and Django as most of everything else has one of these as a dependency. Following that installation, the rest of the software listed in the technology section was installed. With the necessary technology installed, Django was set up to be able to perform all CRUD (Create, Read, Update, Delete) database operations with the data. With Django set up, a basic web interface for the server was created. The next step was to devise a way to obtain course information to use in study group creation. To accomplish this, a python web scraper was created to parse the Georgia Southern course search web page. It performs a search for all courses and saves that information into a CSV (Comma-Separated Values) file. With course information saved into a file, the data is imported into the script and iterated through. With each iteration, the course information is extracted and saved in the database. This gives the database the necessary data to create a basic web interface to use for testing. The web interface is solely used for testing of certain features so it has read-only access to the server. Once the web interface was created and the server was set up, work was able to begin on the app.

The first step when creating the app was to sketch various possible UIs (User Interfaces) and gather feedback. After three versions, a final UI was decided on and coding began on the app. To implement the UI, Android Studio's layout preview tool was used, along with XML for additional fine-tuning. With the UI done, functionality was added with Java and XML attributes. The first feature implemented was user authentication.

To accomplish user authentication, a method was created to asynchronously communicate with the server. When a user enters their credentials on the login screen, they are first validated on client side to ensure that no harmful data is being sent to the server.

Once client-side validation is completed, the user's credentials are stored in a LinkedHashMap object to simplify the process of data transfer. The object is then converted to its raw byte values and stored in an array to be sent to the server. Next, the app opens a secure HTTPS connection with the server to conform to the basic principles of information security; confidentiality, integrity, availability, and non-repudiation. Once a client-server connection is established, the byte array is sent to the server in a POST request. Once the server receives the POST request, if the credentials given to it are valid it sends an HTTP 200 success code along with the user's JWT access token. This token is what is used to access the server after initial authentication to avoid having to send passwords multiple times which carries the risk of being intercepted. Additionally, these tokens expire after a set period of time to prevent their misuse. Once the app receives the token as a JSON response it decodes it and stores it in memory to be used whenever a connection to the server is needed. With the token in memory, the app sends the user to the main menu of the app which is stored as a different activity in the code. However, if an error occurs at any point in the authentication process, a toast notification is displayed asking the user to enter their credentials again.

With user authentication successful, the next step was to connect to the server and display the list of courses to the user. In the course information activity, the user is given textboxes where they can filter which courses they receive information about. This allows filtering by course prefix, course number, both, or neither. Depending on which textboxes are filled, the appropriate server request is prepared and sent, using the token as authentication. The server then responds with a JSON array of courses that fit the criteria specified by the user. This JSON array details the course prefix, course number, course

title, as well as an inner JSON array of section information. This inner array details the section information including section letter, CRN (Course Registration Number), meeting days, meeting time, total seats, instructor, building, and classroom. Once the app receives this JSON response, it parses it to format it in a human-readable format and displays it to the user in a scrollable list. Once the app was able to get course information the next step was to allow users to add courses they are taking to their profile.

To allow association of courses to users, three new activities were created. The first of these allows users to add courses to their profile. To accomplish this, the user is given textboxes to input the prefix and number of the course they wish to add. These values and validated and sent to the server asynchronously in a POST. If a matching course is found, the server stores a relationship between the authenticated user and the specified course. Otherwise, an error message is displayed. Once association of users and courses was added as a feature, the option to disassociate a user and a course was next. To accomplish this, a similar form to course association was created, with the users being asked to input the course prefix and number for the course to be disassociated. This information is validated and used to create an HTTP DELETE request which is then sent to the server to remove the association. The third activity related to course association displays all courses associated with the authenticated user's profile. To accomplish this, when the app is redirected to that activity it immediately sends a request to the server for all course association relationships that involve the authenticated user. When the app receives the JSON response, it formats it in a more human-readable format and displays it to the user in a scrollable list. With these activities complete, users can add courses to their profile, remove courses from their profile, and view courses in their profile.

The next step was to allow the authenticated user to view users with mutual courses. To accomplish this, the app sends a request to the server for all course associations. It then filters the JSON response to only include associations for courses that are associated with the authenticated user. It then uses this to display the authenticated user's list of courses, with the names of classmates displayed below the course.

The next step was to create an activity to displays the authenticated user's study groups. This is accomplished through an HTTP GET request of all study group associations. The JSON response is then filtered to only contain study groups which the authenticated user is a part of. This JSON is then formatted and displayed to the user as a list of study groups followed by the members and their email address.

The final step in the initial development of the app was to allow new users to create an account as until this point new users had to be created on the server. To accomplish this a new activity was created. This activity contains a form which asks users to create a username and password, input their Georgia Southern email address, and rate themselves in various criteria. With this activity complete, the initial development of the app was finished and focus was shifted to development and implementation of the algorithm. Appendix B contains partial source code for the application along with a link to the full source code. Appendix C contains screenshots of the completed application.

Technologies

The backend of the project runs on an Elastic Compute Cloud (EC2) Ubuntu Linux instance hosted on Amazon Web Services (AWS). AWS was chosen as the host for the backend as it is standard in the industry and would provide experience with tools that are

used in the professional world. Additionally, its ease to set up and many pages of documentation allowed a desirable learning curve. On the EC2 instance, the server uses the Django web framework, with a Python 3 dependency. Django was chosen to allow for a faster set up and its built-in features to simplify the app's creation process. Some of the useful Django features used are SQL injection prevention, cross site scripting (XSS) protection, and authentication. By using Django's built-in authentication system, time was not used having to implement a password hash function. MySQL was used as the database due to its ease of use when combined with Django and also its industry-wide prevalence. Amazon Relational Database Service was used to host the database due to its large storage capacity and ease of integration with EC2. Python 3 was used to implement the study group creation algorithm due to its modularity allowing for easy use of libraries as well as its simplified syntax.

The frontend of the project is an Android application developed in Android Studio in Java. Android was chosen as the target Operating System (OS) due to its overwhelming market share of 81.4% at the end of 2016 [2] as well as its simplified development process. The app was developed with a target OS of Android API 19, KitKat, as this gave a large amount of Android API features while still being compatible with most versions of Android.

To allow for communication between the app and the server, the Django REST (REpresntational State Transfer) API was used with JSON (JavaScript Object Notation). Django REST was used to allow for Representational State Transfer of data between client and server. By using RESTful technology, cross platform communication between the Linux web server and the Java Android app was made possible. JSON is used to serialize

data in client-server communications because both target platforms (Python and Java) provide libraries for both encoding and decoding of JSON data.

Database Design

The MySQL database used in the app is a mixture of Django auto-generated tables and custom defined tables. From Django, the User_Tokens and Users tables are used. Custom tables are used for Courses, Course_Sections, User_Courses, Study_Groups, User_Characteristics, and Study_Group_Memberships. The Enhanced Entity-Relationship (EER) diagram for the database is given in Figure 5. As shown in the diagram, a variety of relationships are used throughout. A one-to-one relationship is used between Users and User Tokens as well as between Users and User_Characteristics. By keeping token values as well as user characteristics in separate tables, this data which is accessed regularly is kept more secure and allows for faster processing when received by the scripts. A one-to-many relationship is used between Courses and Course_Sections as every course has at least one section but may have more. There are also two instances of many-to-many relationships. First is the many-to-many relationship between Users and Courses representing which courses each student is taking. A many-to-many relationship is used as a student may take many courses and each course may have many students. To represent this relationship, a lookup table is used with a foreign key to both Courses and Users. The other many-to-many relationship in the database is between Users and Study_Groups representing which study groups a student is part of. A many-to-many relationship is used as every student may belong to many study groups and each study group is comprised of multiple students. To represent this relationship, a lookup table is used with a foreign key to both Users and Study_Groups.

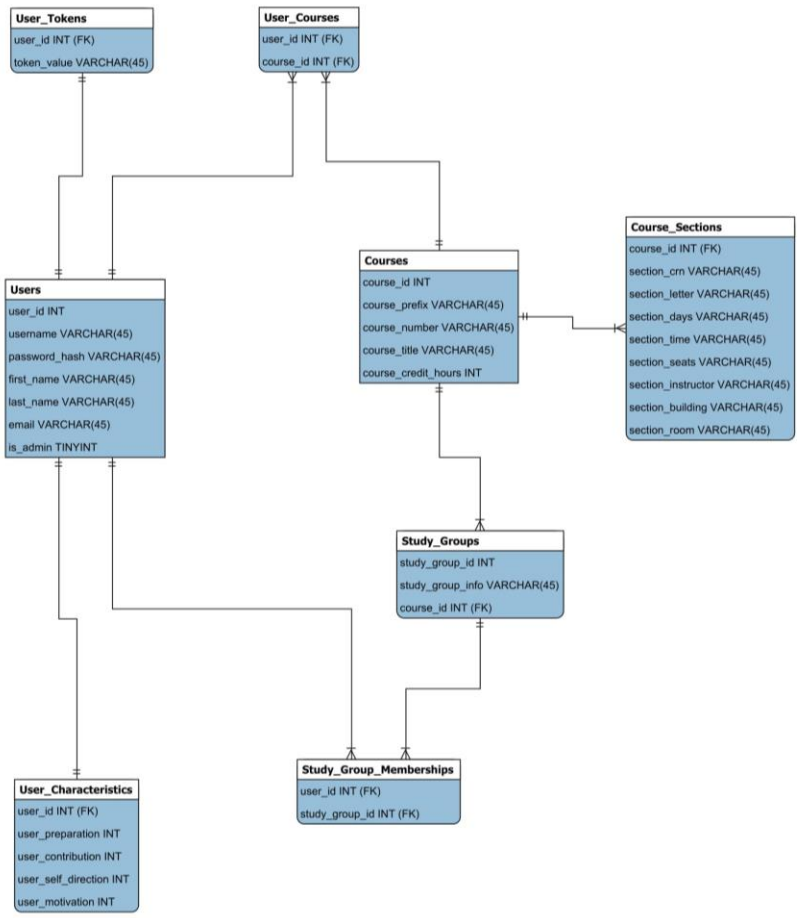


Figure 5: EER Diagram of Database

Results and Analysis

This algorithm was tested with a set of twelve sample students created with randomized values for all four characteristics. Figure 6 shows the mean values for the four characteristics of the resulting study groups. As shown in the figure, the mean value for preparation was 5.25 for two of the groups and 4.75 for the other. Contribution follows the same trend with the three groups having a mean value within one point from each other. Self-direction and motivation had a slightly larger gap between groups but were still within

two points of each other. In this limited test with a size of twelve data points, the algorithm successfully produced heterogeneous study groups with roughly even characteristic values.

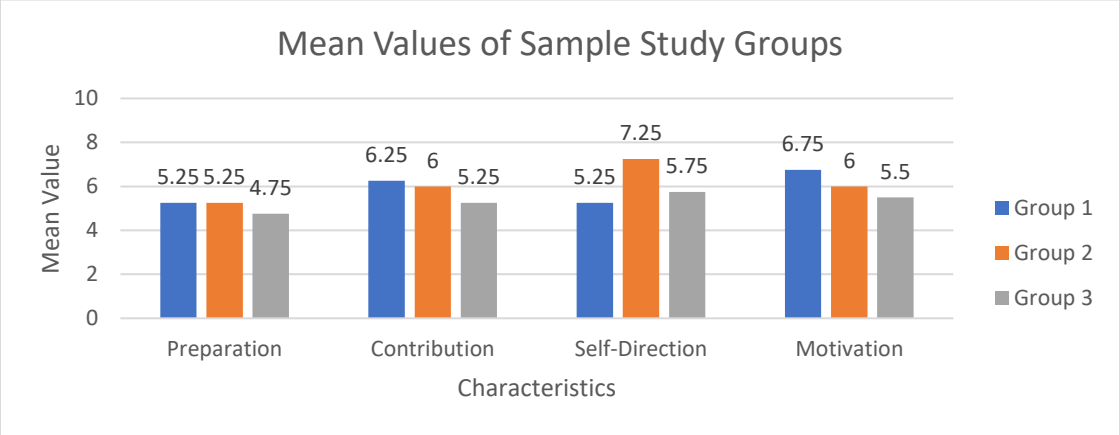


Figure 6: Mean Characteristic Values for Sample Students

Reflection

The entire process of creating an app from start to finish was an extraordinary learning experience. I was exposed to a variety of technologies that I otherwise would not have even heard about. These technologies, namely Django Web Framework and Amazon Web Services, were also useful in later class projects. By learning to use these technologies or my thesis, I was able to take charge in multiple group projects and leverage them to improve overall quality.

In addition to improving the quality of my class projects, completing my thesis has given me experience with technologies and practices used in the industry I will be working in. Full stack development of an Android application required me to set up a backend server. This required me to learn more about Linux, Python, Bash, cloud computing, and more. In addition to the backend server, I also had the opportunity to develop a frontend

Android application that provides an interface for users to interact with. This frontend development required me to dive deeper into the topic of human-computer interaction, something I would not have had the opportunity otherwise. Additionally, I was also able learn how to connect a frontend application to a backend server.

I am extremely grateful for the opportunity to complete an honors thesis. I was able to learn from a project that was not in a classroom setting. I also gained valuable real-world experience that will be invaluable once I enter the work force. This experience is one that I will not soon forget.

Conclusion

Throughout the course of this project, an Android application was developed to help students create study groups. By utilizing basic information about users, the algorithm is able to automatically create effective study groups for students. As shown, these groups are heterogeneous across every characteristic meaning that students have their weaknesses covered.

References

[1] Nan Li, et al, "Watching MOOCS together: investigating co-located MOOC study groups," *Distance Education*, vol. 35, no. 2, p 217-233, April 2014

[2] Idc.com, "Smartphone OS Market Share, 2017 Q1", 2017. [online] Available: https://www.idc.com/promo/smartphone-market-share/os.

[3] Tarun Jain and Mudit Kapoor, "The Impact of Study Groups and Roommates on Academic Performance," *The Review of Economics and Statistics*, vol. 97, no. 1, p 44-54, March 2015

[4] Jessica M. Dennis, Jean S. Phinney, Lizette Ivy Chuateco, "The Role of Motivation, Parental Support, and Peer Support in the Academic Success of Ethnic Minority First-Generation College Students," *Journal of College Student Development*, vol. 46, no. 3, p 223-236, May 2005

[5] Yang-Hsueh Chen and Pin-Ju Chen, "MOOC Study Group: Facilitation Strategies, Influential Factors, and Student Perceived Gains," *Computers and Education*, vol. 86, p 55-70, March 2015

[6] Nadia Rahbek Dyrberg and Claus Michelsen, "Mentoring First Year Study Groups – Benefits from the Mentors' Perspective," *European Journal of Science and Mathematics Education*, vol. 5, no. 1, p 43-54, 2017

[7] Jason M Pittman and Ronald E. Pike, "An Observational Study of Peer Learning for High School Students at a Cybersecurity Camp," *Information Systems Education Journal*, vol. 14, no. 3, p 4-13, May 2016

[8] David R Arendale and Amanda R Hanes, “Adaptability and Flexibility Displayed by Student Leaders of Peer Study Group Sessions,” *Learning Assistance Review*, vol. 21, no. 2, p 9-37, Fall 2016

Appendix A: Interest Survey and Results

Questions

1) What year are you?

2) Are you in the Honors Program?

3) How interested would you be in an app that creates study groups?

4) Any additional comments?

Responses

What year are you?	Are you in the Honors Program?	How interested would you be in an app that creates study groups?	Any additional comments?
Sophomore	Yes	8	
Freshman	Yes	4	
Freshman	Yes	9	
Junior	Yes	7	
Freshman	Yes	8	
Freshman	Yes	8	
Freshman	Yes	7	
Freshman	Yes	9	I feel this will be very useful. Some people find it hard to go for stutter sessions as they feel it degrades them. So an app that helps to create study sessions will be definitely useful.
Sophomore	Yes	1	
Freshman	Yes	8	
Sophomore	Yes	7	

Freshman	Yes	10	
Sophomore	Yes	7	
Freshman	Yes	6	What is a study app exactly?
Freshman	Yes	5	
Freshman	Yes	10	
Sophomore	Yes	7	I think I would have been more interested as a freshman when I didn't know anyone and was in mainly core classes. After that point, you start to get to know people in your major and can make study groups easily.
Freshman	Yes	4	I don't really like study groups but I enjoy apps that connect me with people in the class to ask questions
Freshman	Yes	6	
Sophomore	Yes	8	
Freshman	Yes	10	
Sophomore	Yes	6	
Senior	No	10	I wish we had this while I would still be here.
Sophomore	No	5	
Senior	No	8	
Freshman	Yes	7	
Freshman	Yes	7	
Senior	No	7	
Sophomore	Yes	6	I'd like for there to be profiles of the people so I can see who they have as a professor, class time & also some things about them just to make the best out of a possible study group.
Freshman	Yes	10	
Junior	Yes	8	Integration with other messaging services or social networks would be useful.

Junior	Yes	7	The app should also have a web variant so that it can be accessed on any device.
Junior	No	8	I would love to see an app of this sort of utility, my only issue is the fact that (in my experience) college students would often forget that they had study sessions scheduled and make other plans in their stead. Would an adaptation for built in calendar apps be made, and how often/when would notifications about the study sessions be pushed to the students?
Junior	No	10	I think it would be a good idea because I am too shy to make study groups myself
Senior	Yes	10	
Senior	No	8	
Junior	No	7	
Junior	Yes	7	
Senior	No	10	
Freshman	Yes	7	
Senior	No	9	
Senior	Yes	10	
Freshman	No	9	
Freshman	No	10	
Senior	No	8	
Junior	No	8	
Freshman	Yes	8	
Freshman	Yes	7	
Junior	No	9	
Senior	No	8	
Sophomore	No	7	
Sophomore	No	7	
Freshman	No	9	
Freshman	No	6	
Freshman	No	7	

Sophomore	No	10	
Sophomore	No	3	
Sophomore	No	7	
Sophomore	No	9	
Sophomore	No	8	
Sophomore	No	6	
Junior	No	9	
Junior	No	8	
Junior	No	5	
Junior	No	10	
Junior	No	8	
Senior	No	9	
Senior	No	7	
Senior	No	10	

Appendix B: Partial Source Code

What follows is a portion of the code used for the Android application. This code sample is the Class used for allowing students to add a course to their profile. The complete source code is available at https://github.com/KelvinJRosado/StudyTogether.

```
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.AsyncTask;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;


import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;


import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.Reader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.util.LinkedHashMap;
import java.util.Map;


public class AddCourseActivity extends AppCompatActivity {
```

```
    final String DUPLICATE_COURSE_ADDED =
"{\"non_field_errors\":[\"The fields student, " +
            "courses must make a unique set.\"]}";
    Context thisContext = this;
    //    String username = LoginActivity.username;
    String token = LoginActivity.token;
    int userPK = LoginActivity.userPK;
    int course = 0;
    String coursePrefix = "";
    String courseNumber = "";
    EditText editPrefix, editNumber;
    Button btSubmit, btReturn;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_course);
        setTitle("Add course to profile");
        initGUI();
    }


    @Override
    protected void onResume() {
        super.onResume();
        SharedPreferences sharedPreferences =

PreferenceManager.getDefaultSharedPreferences(getApplicationCo
ntext());
        token = sharedPreferences.getString("token", token);
        userPK = sharedPreferences.getInt("userPk", userPK);
    }


    void hideKeyboard() {
        View view = this.getCurrentFocus();
        if (view != null) {
            InputMethodManager inputManager =
                    (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
```

```
inputManager.hideSoftInputFromWindow(view.getWindowToken(),
0);
        }
    }


    public void onBackPressed() {
        btReturn.performClick();
    }


    void initGUI() {
        btSubmit = (Button)
findViewById(R.id.btAddCourseSubmit);
        btReturn = (Button)
findViewById(R.id.btReturnAddCourse);
        editPrefix = (EditText)
findViewById(R.id.addCoursePrefixEdit);
        editNumber = (EditText)
findViewById(R.id.addCourseNumberEdit);


        btSubmit.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View v) {
                coursePrefix =
editPrefix.getText().toString();
                courseNumber =
editNumber.getText().toString();


                hideKeyboard();


                if (false) {
                    Toast.makeText(thisContext, "Invalid
characters. Try again", Toast.LENGTH_SHORT)
                            .show();
                    return;
                }
```

```
                if (courseNumber.isEmpty() ||
coursePrefix.isEmpty()) {
                    Toast.makeText(thisContext, "Course number
and prefix are required", Toast.LENGTH_SHORT)
                            .show();
                    return;
                }
                //If number and prefix provided
                new GetCourseID().execute();
            }
        });


        btReturn.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View v) {
                Intent mainIntent = new Intent(thisContext,
ContentsActivity.class);
                startActivity(mainIntent);
            }
        });


    }


    private class GetCourseID extends AsyncTask<String, Void,
String> {


        int response = 0;
        StringBuilder stringBuilder;


        @Override
        protected String doInBackground(String... params) {


            String myUrl =
"http://52.2.157.47:8000/courses/?format=json&prefix=" +
coursePrefix
```

```
                    + "&number=" + courseNumber;


            try {
                URL url = new URL(myUrl);
                HttpURLConnection urlConnection =
(HttpURLConnection) url.openConnection();
                urlConnection.setRequestMethod("GET");

urlConnection.setRequestProperty("Authorization", "JWT " +
token);


                try {
                    response =
urlConnection.getResponseCode();
                    InputStream in = new
BufferedInputStream(urlConnection.getInputStream());
                    BufferedReader reader = new
BufferedReader(new InputStreamReader(in));
                    stringBuilder = new StringBuilder();
                    String line;
                    while ((line = reader.readLine()) != null)
{

stringBuilder.append(line).append("\n");
                    }
                    reader.close();


                } catch (Exception e) {
                    e.printStackTrace();
                    return null;
                } finally {
                    urlConnection.disconnect();
                }


            } catch (IOException e) {
                e.printStackTrace();
                return null;
            }
```

```
            return stringBuilder.toString();
        }


        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);


            //Get course id


            if (stringBuilder.toString().equals("[]\n")) {
                Toast.makeText(thisContext, "Course Not found.
Error", Toast.LENGTH_SHORT).show();
                return;
            }


            try {
                JSONArray courseArray = new
JSONArray(stringBuilder.toString());
                JSONObject courseJSON =
courseArray.getJSONObject(0);


                course = courseJSON.getInt("course_id");


                System.out.print(course);


                //Use course id parsed here to add to schedule
                new AddCourse().execute();


            } catch (JSONException e) {
                e.printStackTrace();
            }


        }
```

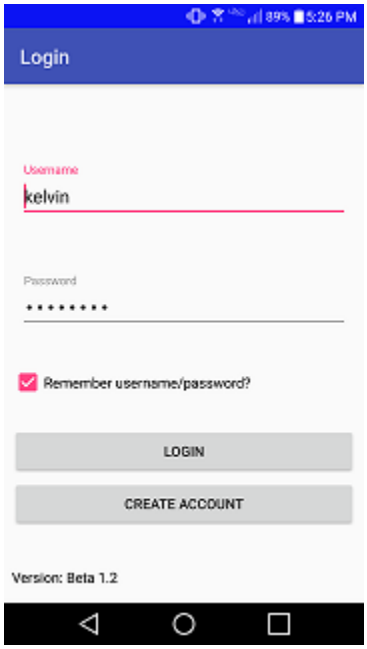
```
    }


    private class AddCourse extends AsyncTask<String, Void,
String> {


        String response = "";


        @Override
        protected String doInBackground(String... params) {


            try {
                String myUrl =
"http://52.2.157.47:8000/addcourse/";
                URL url = new URL(myUrl);


                Map<String, Object> mapParams = new
LinkedHashMap<>();
                mapParams.put("student", userPK);
                mapParams.put("courses", course);


                //Make StringBuilder object of POST data
                StringBuilder postData = new StringBuilder();
                for (Map.Entry<String, Object> param :
mapParams.entrySet()) {
                    if (postData.length() != 0)
                        postData.append('&');//Seperate args
with & char
                    //Append username, password, and email and
encode

postData.append(URLEncoder.encode(param.getKey(), "UTF-8"));
                    postData.append('=');

postData.append(URLEncoder.encode(String.valueOf(param.getValu
e()), "UTF-8"));
                }
```

```
                //Get postData as bytes
                byte[] postDataBytes =
postData.toString().getBytes("UTF-8");

                //Open connection
                HttpURLConnection connect =
(HttpURLConnection) url.openConnection();

                //Write username and password and get token
                connect.setRequestMethod("POST");
                connect.setRequestProperty("Content-Type",
"application/x-www-form-urlencoded");
                connect.setRequestProperty("Content-Length",
String.valueOf(postDataBytes.length));
                connect.setRequestProperty("Authorization",
"JWT " + token);
                connect.setDoOutput(true);

connect.getOutputStream().write(postDataBytes);

                Reader input = new BufferedReader(new
InputStreamReader(connect.getInputStream(), "UTF-8"));

                //Get response as String
                StringBuilder sb = new StringBuilder();
                for (int c; (c = input.read()) >= 0; )
                    sb.append((char) c);

                response = sb.toString();

                connect.disconnect();

            } catch (IOException e) {
                e.printStackTrace();
            }
```

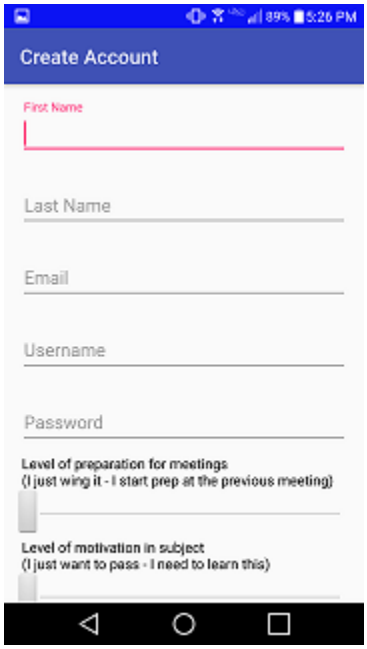
```
            return null;


        }


        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);


            if (response.contains(DUPLICATE_COURSE_ADDED)) {
                Toast.makeText(thisContext, "Course already
exists on your schedule",
                        Toast.LENGTH_SHORT).show();
                return;
            }


            Toast.makeText(thisContext, "Course successfully
added", Toast.LENGTH_LONG).show();


        }
    }
}
```

Appendix C: Screenshots

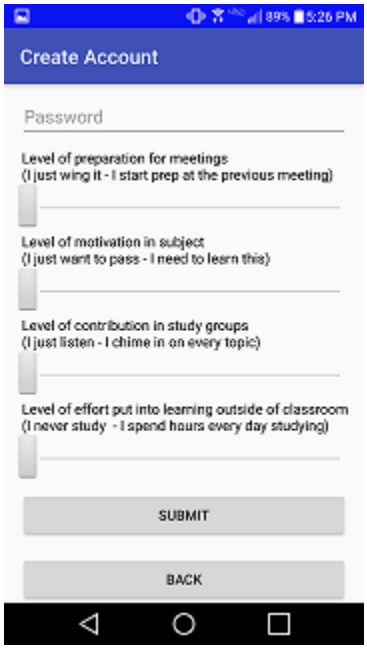
1: Start Screen



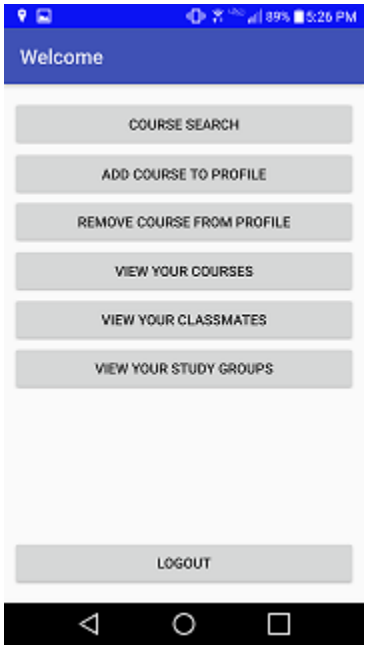
2: Top of Account Creation Screen



3: Bottom of Account Creation Screen

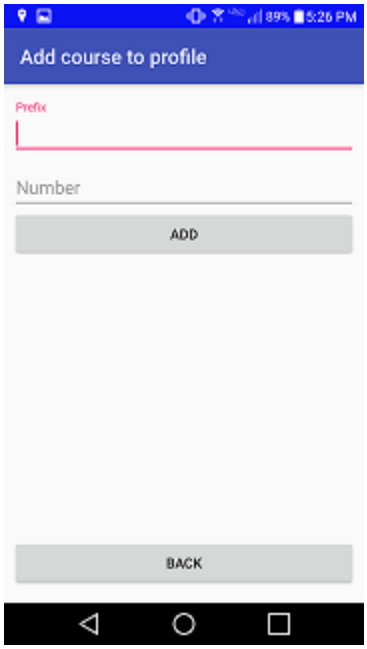


4: Main Menu



5: Course Search

6: Adding a course



7: Removing a Course

8: Viewing of added courses

9: Viewing of Study Groups